DOCUMENT RESUME

ED 326 237 IR 053 335

AUTHOR McFadden, Fred R.
TITLE A Methodology for Distributing the Corporate Database.
PUB DATE May 90
NOTE 12p.; In: Proceedings of the 1990s Information Resources Management Association International Conference (Hershey, PA, May 14-16, 1990); pp.55-65.
PUB TYPE Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Networks; *Computer Software Development; *Database Management Systems; Data Processing; *Information Retrieval; Management Information Systems; Methods Research; *Programing
IDENTIFIERS Relational Approach (Information Retrieval); *Relational Models

ABSTRACT

The trend to distributed processing is being fueled by numerous forces, including advances in technology, corporate downsizing, increasing user sophistication, and acquisitions and mergers. Increasingly, the trend in corporate information systems (IS) departments is toward sharing resources over a network of multiple types of processors, operating systems, and applications. A key design issue that arises in distributed systems concerns is how to optimally distribute data among the nodes of a network. This paper describes the problem of database distribution and proposes a methodology for the distribution of data that maximizes local references to data. The methodology considers the frequency of accesses to data from each of the nodes, and specifically accounts for the difference in cost between retrievals and updates in a distributed network. The methodology can be applied using hand calculations for smaller problems and can be also implemented in software for larger problems. (12 references) (Author/SD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

A Methodology for Distributing the Corporate Database

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Fred R. McFadden
Professor of Information Systems
College of Business and Administration
University of Colorado at Colorado Springs
Colorado Springs, CO 80933-7150

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Fred R.McFadden
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ABSTRACT

A key design issue that arises in distributed systems concerns how to optimally distribute data among the nodes of a network. This paper describes the problem of database distribution and proposes a methodology for the distribution of data that maximizes local references to data. The methodology considers the frequency of accesses to data from each of the nodes, and specifically accounts for the difference in cost between retrievals and updates in a distributed network. The methodology is easily applied using hand calculations for smaller problems and can be easily implemented in software for larger problems

1.Introduction

The trend to distributed processing is being fueled by numerous forces: advances in technology, corporate downsizing, increasing user sophistication, and acquisitions and mergers, among others. Increasingly the trend in corporate IS departments is toward sharing resources over a network of multiple types of processors, operating systems, and applications [7]. According to a recent survey [6] over 50 percent of companies in the U.S., Japan, and France have moved, or are moving, to decentralized information systems structures.

A major component of this trend is the evolution of distributed database management. The corporate database, previously centralized, may now be distributed (all or in part) over the nodes of a network. Often one (or more) of the nodes functions as a database server, providing database management services for other nodes in the network.

A key design issue that arises in distributed systems concerns how to distributed data among the nodes of a network. The choice of distribution strategy often has a major impact on performance, reliability, data availability, and other key factors. This paper describes the problem of database distribution and proposes a methodology for the distribution of data that will maximize local references to data. According to Rothnie and Goodman [10], in a well-designed distributed database, "90 percent of that data should be found at the local site, and only 10 percent of the data should be accessed on a remote site.

There is a rather extensive literature on the problem of database distribution design. For example, in an early paper Chang and Cheng [4] propose a methodology for decomposing a relation into fragments. However, they do not suggest a methodology for the optimal distribution of

fragments. Ceri, Navathe, and Wiederhold [1] propose a model for the optimal horizontal fragmentation of relations in a distributed database. While comprehensive, their model is based on the use of linear integer programming and appears difficult to apply in practice. Navathe, Ceri, Wiederhold, and Dou [9] have studied the application of vertical fragmentation to distributed databases but do not consider the more common case of horizontal fragmentation. Ceri and Pelagatti [2] provide a good overall discussion of distributed database systems.

In a more recent paper, Ceri, Pernici, and Wiederhold [3] discuss a methodology for the horizontal distribution of data. They describe DATAID-D, a computer-based methodology for the top-down design of data distribution. The authors define several tables that are used to store information that define the frequency of access to data by the various applications at each node or site. Similar tables are used, in modified form, in the present paper. However, the methodology proposed in this paper extends the work of these authors in several directions:

1. It includes a formal statement of the problem of database distribution, in terms of entities, relationships, nodes, and applications.
2. It specifically accounts for multiple accesses to an entity for each activation of an application, which occurs for example when there is a 1:M association between two entities used by that application.
3. It specifically accounts for the difference in cost between retrievals and updates in a distributed network.
4. It provides a straightforward method for evaluating the proposed replication of fragments at more than one location.

The organization of this paper is as follows: Section two provides a narrative description of the problem of distribution design, in which several key terms are described. Section three provides a formal statement of the distribution problem, then develops a methodology for the optimal distribution of data to maximize locality of references. Section four then presents an extended example for a manufacturing company to illustrate the application of the methodology. Section five discusses the problem of secondary distribution (or replication) of data, and applies the methodology of the previous section to this problem. Finally, section six summarizes the results and suggests future research directions.

2. Distribution Design

Database design is generally considered to consist of four phases: requirements analysis, conceptual design, logical design,and physical design [8]. For organizations that use distributed processing an additional design step, called distribution design, is required. Distribution design involves the partitioning of logical data into separate fragments, and the assignment of each fragment to one or more nodes or locations [5].

2.1 Fragments

In this paper we assume the existence of a relational database system in discussing distribution design. Although other types of databases (for example, hierarchical and network) may also be distributed, the problem is conceptually much simpler for relational databases. Also, relational databases are evolving as an industry standard for a majority of applications.

A fragment is a partition of a relation (thus it is a subrelation). Two types of fragments may be considered, horizontal and vertical.

Horizontal fragments are derived by selecting tuples from a relation. For example, suppose we define the relation EMPLOYEE (EMPLOYEE_NUMBER, EMPLOYEE_NAME, DEPARTMENT). Then one fragment is defined by the following SQL statement:

```
SELECT EMPLOYEE_NUMBER,
EMPLOYEE_NAME, DEPARTMENT
FROM EMPLOYEE
WHERE DEPARTMENT = 'MARKETING'
```

Vertical fragments are derived by selecting attributes from a relation, for example, a vertical fragment of the employee relation could be defined as follows:

```
SELECT EMPLOYEE_NAME, DEPARTMENT FROM
EMPLOYEE
```

Mixed fragments may be defined by combining the horizontal and vertical fragmentation operations (the equivalent of SELECT and PROJECT in relational algebra).

In this paper we consider only the case of horizontal fragmentation, for two reasons:

1. It appears by far to be the most common case.
2. It is conceptually the simplest to manage (vertical and mixed fragmentation are much more difficult).

One limitation of the fragmentation allocation approach described here is that foreign keys are not considered. Foreign keys define relationships between relations that are often of great importance to the efficient execution of an application.

2.2 Objectives of Distribution Design

According to Ceri and Pelagatti [2, pp. 69-70], there are several possible objectives in distribution design:

1. Processing locality — placing data as close as possible to the applications which use them.
2. Availability and reliability of data — quicker access and backup provided by multiple copies (in the case of replicated data).
3. Workload distribution — distribute the workload over the nodes of a network, to maximize the use of computers at each site.
4. Storage costs and availability — minimize the costs ofstorage by optimizing the use of storage devices.
5. Minimizing data communications costs.

It is virtually impossible to derive a model (or methodology) that simultaneously considers all of these criteria. Instead, it is necessary to select the most important criterion and to treat the remaining features as constraints. This approach provides insight into the nature of the real-world problem of data allocation.

The methodology developed in this paper seeks to maximize processing locality. The problem may therefore be stated as follows: For each relation in the database, select a partitioning strategy that partitions the relation into horizontal fragments, and then assign each fragment to one (or more) nodes or sites (of course, one possible strategy would be to not partition the relation at all but rather to assign it to a single site). Defining a partitioning strategy is equivalent to identifying a set of qualifiers (or predicates) that are used in selecting the rows of the relation that are assigned to each partition. Each strategy assumes that there are computer systems and staff at each location capable of handling the basic transactions of the business.

Distribution design is performed in two phases, primary distribution and secondary distribution.

Primary distribution is the process of partitioning a relation into fragments, and the assigning of each fragment to a unique site. Secondary distribution is the process of replicating one or more fragments at multiple locations.

3. Problem Formulation

Following is a statement of the problem of the primary distribution of data in a distributed network.

1. There are M nodes N_j, j = 1 to M in a distributed computing network.
2. There are R entities E_k, k = 1 to R in a database.
3. Each entity is to be partitioned into P fragments F_t, t = 1 to P where $1 \leq P \leq M$.

Each fragment is then to be assigned to a unique node in the network (the possibility of assigning a fragment redundantly to more than one node is discussed later).

4. For each entity there are one or more partitioning strategies. Let S_n, n = 1 to K

designate the strategies to be considered for the entity E_k. Let $F_j kn$ designate the fragment of entity E_k that is assigned to node j by strategy S_n.

5.There are T applications A_i, i = 1 to T that access data in the network. Each of the applications may be activated from any of the nodes.

6.An application may access zero, one, or several occurrences of a given entity each time it is activated. Let $a_i k$ represent the *average* number of occurrences of entity k that are referenced by application i each time that it is activated. This data is represented in the logical access map for eachapplication.

The distribution problem may now be stated as follows: for each entity (E_k), select the partitioning strategy S_n that will maximize the expected number of local references to the fragments for that entity.

3.1 Information Requiremer.ts

A methodology to solve this problem requires that the following information be available.

1. The entities that are accessed by each application, and the number of accesses to each entity. This information is provided in the form of logical access maps.
2. The frequency that each application is issued from each of the nodes. Let $U_i j$ denote the frequency that application i originates at node j per unit time. This data (which must be estimated if not already available) will be presented in a table called the application frequency table.
3. The relative frequency that an application which is issued at a given node will access a fragment that is stored at another node. Let $f_i jkn$ denote the relative frequency that application i is activated at node j and accesses a fragment at node k for partitioning strategy S_n. This data will be presented in a table called the application incidence table.

The expected number of local accesses

Figure 1: Distributed Processing System

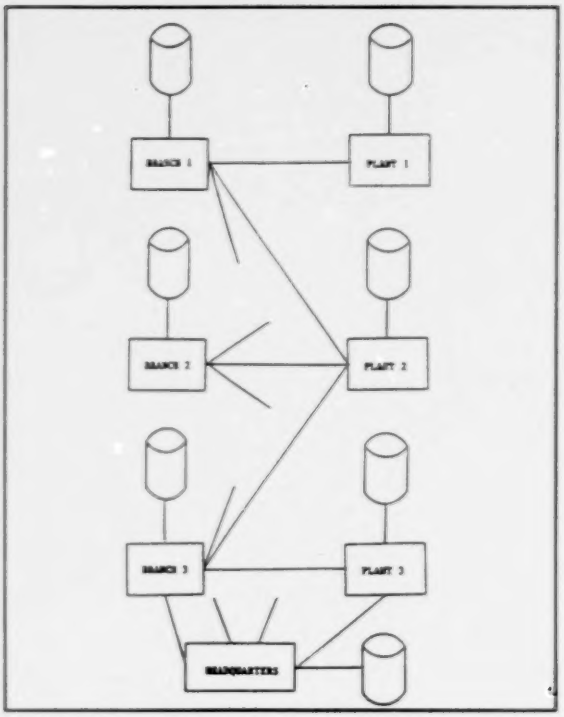
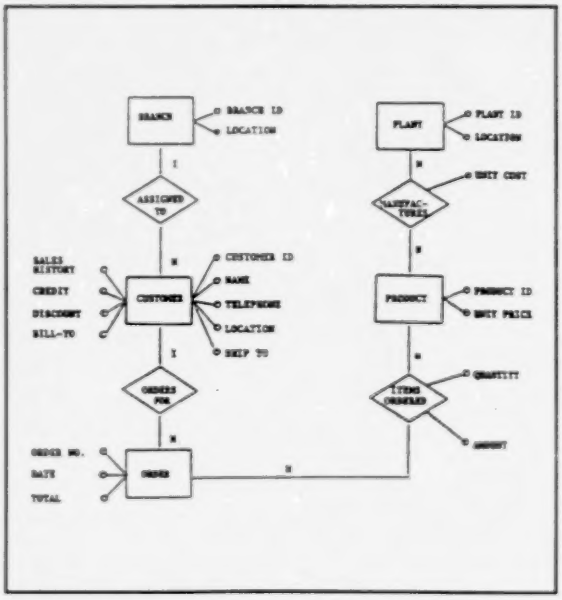


Figure 2: Global Schema for Manufacturing Company

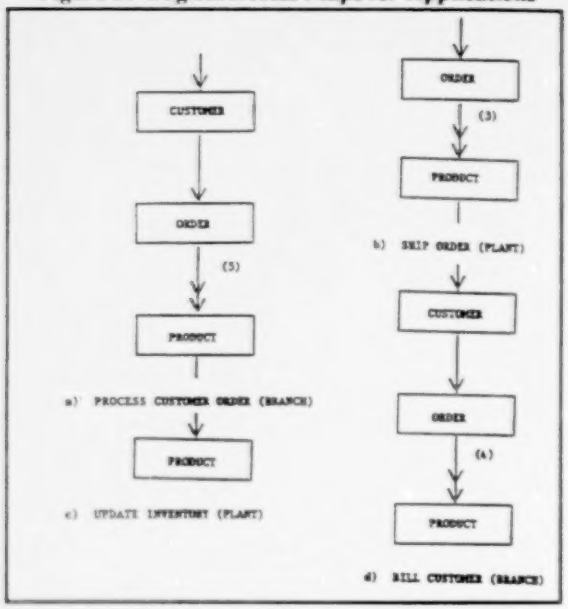


for an entity k withstrategy S_n is thus given by:

$$E_k(n) = \sum_{i=1}^{T} \sum_{j=1}^{M} U_i j a_i k f_i jkn(1)$$

In the above expression the first sum

Figure 3: Logical Access Maps for Applications



ranges over all applications, and the second sum ranges over all nodes. The subscript j appears twice in f_{ijjn} since this represents local accesses — that is, applications that are activated at node j and access a fragment stored at that same site. These terms correspond to diagonal elements in the application incidence table.

3.2 Retrievals Versus Updates

Equation (1) computes the expected number of local accesses (or references) to an entity. However, it makes no distinction between the types of accesses to that entity. In particular, it does not distinguish between retrieval accesses and update accesses. Generally, in a distributed database network update accesses are more costly than retrieval accesses, since more control messages must be exchanged when updating at a remote node. Using equation (1) might lead to a partitioning strategy that would maximize local retrieval accesses but would force extensive remote updating of the database.

To correct for this problem, a weighting factor W_i is introduced for each application i. The factor W_i is determined by two other factors:

1. The proportion r_i of update access to total accesses for application i ($0 \le r_i \le 1$).
2. The proportion C_i which represents the ratio of the cost of an update access to a retrieval access for application i (this proportion may be constant, or it may vary from one application to the next).

The weighting factor W_i may then be defined as follows:

$$W_i = 1 + r_i (C_i - 1) \qquad (2)$$

The effect of this factor may be seen as follows. Suppose that $c_i = 2$ (updates are twice as costly as retrievals). If $r_i = 0$ (all references are retrievals only) then $W_i = 1$. If $r_i = 1$ (all references are updates) then $W_i = C_i = 2$. In partial updates ($0 < r_i < 1$), $1 < W_i < 2$.

Introducing the weighting factor Wi into equation (1) yields the following:

$$E_k(n) = \sum_{i=1}^{T} \sum_{j=1}^{M} W_i U_r a_i k f_{ijn} \qquad (3)$$

Table 1: Application Frequency Table (Activations/Week)

APPLICATION TYPE (i)	ORIGINATES AT (j)					
	BRANCH			PLANT		
	1	2	3	1	2	3
1. PROCESS ORDER	1000	2000	1500	0	0	0
2. SHIP ORDER	0	0	0	2500	2000	1500
3. BILL CUSTOMER	1500	3000	2000	0	0	0
4. UPDATE INVENTORY	0	0	0	4000	3000	3000

Having computed the expected number of weighted local accesses $E_x(n)$ using the above formula for each partitioning strategy, the last step is to select the strategy that maximizes $E_x(n)$. That is,

$$n_x^* = \max_{\text{all } n} E_x(n) \quad (4)$$

The use of these equations is illustrated by means of an example below.

4. Example

To illustrate the methodology just described we will consider the example of a manufacturing company (see Figure 1). The company has three manufacturing plants that are located in different regions of the country. Customers are served by three branch sales offices that are also located in separate geographical regions. At the present time all computer operations are centralized at corporate headquarters which are located in the Midwest.

To improve its responsiveness to its customers and to stimulate local decision making, the company has decided to distribute its computer operations. A wide-area network will link the computers at each of the seven locations (see Figure 1). The company is deciding on a strategy for distributing the corporate data that is now centralized at headquarters.

A global schema for the corporate database is shown in Figure 2, in the form of an entity-relationship diagram. The company has several product lines. Some of these are produced at only one plant, while others are manufactured at two of the plants (none of the products are produced at all three plants).

Customers of the company are mostly wholesale firms and distributors. Each customer may of course have offices in several geographical locations. In the past each customer has been assigned to one of the branch offices, so that office would become familiar with the customer's needs. The company has not yet decided whether to continue this policy when the data processing system is distributed.

4.1 Logical Access Maps

There are four major applications that will have the greatest impact on the placement of data: Process Customer Order (at branches), Ship Customer Order (at plants), Update Inventory (at plants), and Bill Customer (at branches). Logical access maps for these applications are shown in Figure 3. Each map shows the sequence of accesses to entities for that application, and (for one-to-many relationships) the average number of instances accessed for each activation of the

Table 2: Application Incidence Table (Customer Entity)

		APPLICATION TYPE (i)					
		PROCESS CUSTOMER ORDER			BILL CUSTOMER		
PARTIONING STRATEGY(n)	FRAG-MENT(k)	ORIGINATES AT BRANCH (j)			ORIGINATES AT BRANCH (j)		
		1	2	3	1	2	3
"HOME-OFFICE" ADDRESS	1	50	20	20	60	25	20
	2	25	55	20	20	65	20
	3	25	25	60	20	10	60
"SHIP-TO" ADDRESS	1	80	10	5	50	25	15
	2	15	75	20	20	60	15
	3	5	15	75	30	15	70
"BILL-TO" ADDRESS	1	45	25	20	75	10	10
	2	25	55	30	15	80	15
	3	30	20	50	10	10	75

application.

Following is a brief description of each of these applications.

1. Process Customer Order: A new customer order is received at one of the branches. Customer credit information is checked, and then a new order is created. Each order specifies an average of 5 products.The inventory level for each item is checked to determine whether the item is in stock or will have to be backordered, and an acknowledgement is sent to the customer. After order entry is completed the order is transmitted by electronic communications to one or more of the plants to be filled (no inventories are maintained at the branches).
2. Ship Customer Order: A customer order is processed at a plant and the items are shipped to the customer. An average of three items is shipped from a plant for a given order. The inventory levels for these items are reduced when they are shipped (if an item is out of stock, it is backordered). Detail concerning items shipped is transmitted back to the branches for billing purposes.
3. Bill Customer: The branch bills the customer for items shipped. The Product records must be accessed for this operation to obtain pricing information. An average of 4 product records is accessed for this operation (this number is less than the average order size because of items backordered and shipped separately).
4. Update Inventory: The Product file is used for a variety of applications in production scheduling and inventory control at the plants. Update Inventory is a composite of those applications, indicating the usage of the Product file for these transactions.

4.2 Application Frequency Data

The Application Frequency Table for the manufacturing company is shown below (see Table 1). This table shows the average number of activations of each application at each location per week (U_{ij}). The headquarters location is not shown in the table because none of the four applications is activated at that site. The data in Table 1 is based on historical records, with a correction for expected growth.

4.3 Partitioning Strategies: Customer Entity

All of the data are currently located at corporate headquarters. How should the data be distributed among the various sites? The company is currently considering several possible strategies.

The Customer entity should clearly be distributed among the branches, since it is not used at the plants (this is consistent with company policy to have one branch deal with each customer). The company is considering three strategies for assigning customers to branches:

1. Assign to the branch which is closest to the customer's headquarters location.
2. Assign to the branch which is closest to the customer's primary "ship-to" location (this will tend to localize references for the Process Customer Order application).
3. Assign to the branch which is closest to the customer's "bill-to" location (this will tend to localize references in the Bill Customer application). The bill-to location may or may not be the headquarters location.

The Application Incidence Table for the Customer entity for these three strategies is shown in Table 2. Each entry in the table shows

for each application that is activated at a given site, and for each partitioning strategy, the relative frequency of accesses to a fragment of the Customer entity (denoted f_{jkn}). For example, consider the second partitioning strategy ("ship-to" address: n=2). Also, consider the Bill Customer (i=2) application which is activated at Branch 1 (j=1). Then according to Table 2, 50 percent of the references will be local (that is, to a customer record which is in the fragment located at Branch 1, so that k=1). On the other hand, 20 percent of the references will be at branch 2, and 30 percent will be at branch 3.

For primary partitioning of data only the diagonal elements (j=k) are of interest, since they represent local references. However, the off-diagonal elements provide important information about the possible replication of data at sites other than the primary site.

4.4 Distributing Customer Data

We now use the above equations to determine the optimal primary allocation of customer data. First, we use equation (2) to compute the weighting factors. We assume that $C_i = 2$ (in actual practice, updates would probably be more than twice as costly as simple retrievals). For the Process Customer Order application, most accesses to the Customer entity are retrieval only ($r_1 = 0$). Thus the weighting factor is

$$W_1 = 1 + O\,(2 - 1) = 1$$

For the Bill Customer application, most references are updates (customer accounts receivable are updated). Thus (since $r_2 = 1$):

$$W_2 = 1 + 1\,(2 - 1) = 2$$

For each of the above applications, the average number of accesses to the Customer entity each time the application is used is $a_1 1 = a_1 2 = 1$ (see Figure 3). Thus for the first partitioning strategy (n = 1) in Table 2 (locate at branch nearest corporate headquarters), applying Equation 3 yields the following:

$$E1(1) = 1(1000)(1)(0.50) + (1)(2000)(1)(0.55) + 1(1500)(1)(0.60) + 2(1500)(1)(0.60) + 2(3000)(1)(0.65) + 2(2000)(1)(0.60) = 10,600$$

Applying the same calculations for partitioning strategies 2 and 3 yields the following results

$$E_1(2) = 11,325;\ E_1(3) = 12,350$$

Thus applying equation (4), the optimum strategy is $n_1^* = 3$.

Partitioning strategy 3 should be used, and the Customer data should be distributed among the branches according to the customer "bill-to" address.

4.5 Distributing Product Data

For the Product entity the distribution decision is more complex than for customer data, for three reasons:

1. Product data is used at the plants as well as at the branches.
2. There are multiple references to a product for each transaction. For example, for Process Customer Order the average number of references is $a_1 2 = 5$ (see Figure 3).
3. All four application types reference the product entity (as shown in Figure 3).

The weighting factor for each of the four applications is computed below (it is again assumed that $c_i = 2$ for each application).

1. Process Customer Order: $r_1 = 0$, so $w_1 = 1$.
2. Ship Customer Order: $r_2 = 1$, so $w_2 = 2$.
3. Bill Customer: $r_3 = 0$, so $w_3 = 1$.
4. Update Inventory: $r_4 = 0.75$, so $w_4 = 1 + 0.75$

$(2 - 1) = 1.75$ Two partitioning strategies are being considered for the Product entity (see Table 3).

1. Locate each occurrence at that branch where it is most frequently referenced. This strategy will tend to localize accesses for the Process Customer Order and Bill Customer applications, but will require remote updating from the plants for the Ship Customer Order and Update Inventory application.

Table 3: Application Incidence Table (Product Entity)

POSITIONING STRATEGY (n)	FRAGMENT (k)	APPLICATION TYPE (i)											
		PROCESS ORDER			SHIP ORDER			BILL CUSTOMER			UPDATE INVENTORY		
		ORIGINATES AT (j)			ORIGINATES AT (j)			ORIGINATES AT (j)			ORIGINATES AT (j)		
		B1	B2	B3	P1	P2	P3	B1	B2	B3	P1	P2	P3
LOCATE AT BRANCH WHERE MOST USED	B1	40	25	20				40	25	20			
	B2	30	50	30				30	50	30			
	B3	30	25	50				30	25	50			
LOCATE AT PRIMARY PRODUCTION FACILITY	P1				60	15	10				65	15	15
	P2				20	70	20				20	70	10
	P3				20	15	70				15	15	75

2. Locate each occurrence at the primary production facility for that product. This strategy will tend to localize the update operations described above, but will require remote access from the branches for Process Customer Order and Bill Customer.

Following are the calculations using Equation (3) for each of these strategies.

1. Locate at Branch where Most Used (n = 1).

$$E_2(1) = 1(1000)(5)(0.40) + 1(2000)(5)(0.50) + 1(1500)(5)(0.50) + 1(1500)(4)(0.40) + 1(3000)(4)(0.50) + 1(2000)(4)(0.50) = 23,150$$

2. Locate at Primary Production Facility (n = 2).

$$E_2(2) = 2(2500)(3)(0.60) + 2(2000)(3)(0.70) + 2(1500)(3)(0.70) + 1.75(4000)(1)(0.65) + 1.75(3000)(1)(0.70) + 1.75(3000)(1)(0.75) = 35,863$$

Thus it is better to distribute the Product data to the plants, largely because of the heavy updating workload that occurs there (if the weighting factor w_i is not used, the Product data will be distributed to the branches and much remote updating will be required).

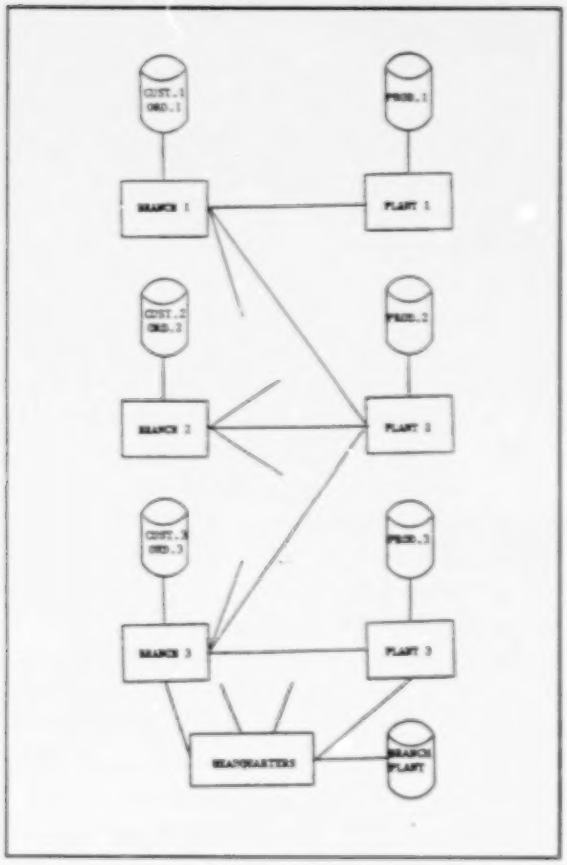
4.6 Distributing Order Data

Now that the Customer and Product data have been distributed, it is necessary to establish a basis for distributing the Order data. Customer and Product are "permanent" entities, while Order is a transaction entity that links customers and products (see Figure 2). Our intuition tells us that each customer order will be stored at the branch where the "owning" customer occurrence is stored. There is a formal reason why this strategy will in fact be used: The association from Customer to Order is 1:M, while the association from Product to Order is M:N. Thus there is a unique assignment of an order occurrence to a customer occurrence (but not to a product occurrence).

In general, where there is a 1:1 or 1:M relationship between a permanent entity and a transaction entity, the transaction entity will be fragmented using the same strategy as for the permanent entity (assuming there is only one such relationship). If there are two (or more) such relationships then the method of the previous sections will be required to determine a distribution strategy.

If the order data is stored at the branches, how will the association between the Order and Product entity be established? By transmitting the necessary order data to the plants as part of the Process Customer Order application, as de-

Figure 4: Distribution of Corporate Data



scribed earlier.

The primary distribution of data for the manufacturing company is shown in Figure 4. In this diagram CUST.1 indicates the fragment of the Customer entity assigned to branch 1, ORD.2 the Order fragment assigned to branch 2, PROD.3 the fragment of the Product entity assigned to branch 3, and so on. The Branch and Plant entities are maintained at company headquarters since they provide the basis for summary information required at that location.

5. Secondary Distribution (Replication)

All of the (previously centralized) data have now been distributed nonredundantly to the plant and branch locations. The next issue to be decided is whether any of the fragments should be replicated at any of the sites. The purpose of replication (if used) would be to further increase the number of local accesses to this data.

A proposed strategy for replication of data can be evaluated by using a variation of equation (3). Instead of summing over the diagonal elements of the application incidence table (Tables 2 and 3), it is necessary to sum over selected non-diagonalelements of this table. We illustrate this process below.

In the primary distribution of data, all Product fragments are stored at the plants (see Figure 4). Of course, this means that the branches must access this data remotely for two major applications: Process Customer Order and Bill Customer (these accesses are all retrieve only, since the branches do not update product data). Suppose that in an attempt to eliminate remote accesses, the company considers storing a full copy of the Product entity at Branch 2. We evaluate this proposal by comparing the remote accesses eliminated, to the additional remote accesses required to update this new database component.

With a copy of the Product entity at branch 2, all remote accesses to this data at the plant are of course eliminated. This includes accesses from branch 2 to plant 1 (B2-Pl), B2-P2, and B2-P3. We apply equation 3, using only those terms that apply.

a. Process Customer Order
Weighted accesses: 1(2000)(5)(1.0) = 10,000

b. Bill Customer:
Weighted accesses: (1)(3000)(4)(1.0) = 12,000

Total remote accesses avoided: 10,000 + 12,000 = 22,000. Unfortunately, the copy of the Product entity at B2 will require updating for all product data performed at the plants.

a. Ship Order
Weighted accesses: 2 (2500 + 2000 + 1500) 3 = 36,000b.Update Inventory
Weighted accesses: 1.75 (4000 + 3000 + 3000) = 17,500
Total new accesses: 36,000 + 17,500 = 53,500

Thus the total number of new accesses

required far exceeds the number avoided, and storing a copy of the Product entity at B2 is not warranted. In fact, storing a full copy of the product data at any of the locations (branches or plants) is not warranted.

6. Summary

A detailed statement of the problem of distributing data among the nodes of a network has been presented. After a formal statement of the problem was given, a detailed methodology was proposed that allows us to evaluate various partitioning strategies with the objective of maximizing local accesses to the data. This methodology includes accounting for the increased overhead associated with remote updates, compared with retrieval accesses. An extended example was used to illustrate the methodology.

This research may be extended in several directions, as suggested below.

1. The methodology needs to be tested empirically in various real-world environments.
2. The problem of data collection and/or estimation (for example, Tables 1, 2, and 3) should be investigated.
3. The conditions under which replication of data is justified should be explored.
4. The methodology could be extended to include additional criteria such as response times, reliability, and so on.
5. Perhaps an expert system could be developed that would incorporate the methodology but simplify the user interface.

References

[1]Ceri, S., S. B. Navathe, and G. Wiederhold, "Distribution Design of Logical Database Schemas," IEEE Trans. Software Engineering, 9, 4 (July 1983), pp. 487-504.

[2]Ceri, S. and G. Pelagatti, Distributed Databases: Principles and Systems, McGraw-Hill, New York, 1984.

[3]Ceri, S., Pernici, B., and G. Wiederhold, "Distributed Database Design Methodologies," Proceedings of the IEEE, 75, 5 (May 1987), pp. 533-546.

[4]Chang, S. K. and W. H. Cheng, "A Methodology for Structured Database Decomposition," IEEE Trans. Software Eng., 6, 2 (March 1980), pp. 205-218.

[5]Date, C. J., An Introduction to Database Systems, Volume II, Addison-Wesley, 1983, pp. 297-298.

[6]Kerr, Susan, "IS Trims Down," Datamation, 34, 19 (October 1, 1988), pp. 46-49.

[7]Liebenau, Carol, "Weaving Seamless Computer Systems," Datamation, 34, 17 (September 1, 1988), pp. 59-66.

[8]McFadden, Fred R. and Jeffrey A. Hoffer, Data Base Management, 2nd edition, Benjamin/Cummings, Menlo Park, CA, 1988, pp. 240-242.

[9]Navathe, S. B., Ceri, S., Wiederhold, G., and J. Dou, "Vertical Partitioning Algorithms for Database Design," ACM Transactions on Database Systems, 9, 4 (December 1984), pp. 680-710.

[10]Rothnie, J. B. and N. Goodman, "A Survey of Research and Development in Distributed Database Management Systems," Proceedings of 3rd International Conference on Very Large Data Bases, 1977.

[11]Segev, A., "Algorithms for Distributed Data Processing," Computers and Operations Research, Volume. 14, No.2, 1987, pp. 145-162.

[12]Teorey, Toby J., Chaar, Jarir, Olukotun, Kunle, and Amjad Umar, "Allocation Methods for Distributed Database," Database Programming and Design, 2,4 (April 1989), pp.34-45

Fred McFadden is Professor of Information Systems at the University of Colorado at Colorado Springs. He received a Ph.D. in Industrial Engineering from Stanford University. He is co-author of the textbook *Data Base Management* and has published numerous articles in the fields of information systems and production/operations management. Professor McFadden's primary research interests are in the fields of distributed database management, decision support systems, and knowledge-based systems.

END

U.S. Dept. of Education

Office of Education
Research and
Improvement (OERI)

ERIC

Date Filmed

March 29, 1991